

I hereby certify that this correspondence is being deposited as
express mail # EL605496177 US
with the United States Postal Service in an envelope addressed
to: Commissioner for Patents, P. O. Box 1450, Alexandria, VA.
22313-1450 on APRIL 2, 2004

By [Signature]
Date of Signature APRIL 2, 2004

Wireless Receiver Code Download and Boot Sequence

Field of the Invention

This invention relates to an apparatus and a method for the transmission and reception of code for a wireless receiver, including a booting mechanism for a Central Processor Unit (CPU).

Background of the Invention

There are several mechanisms used for providing start-up instructions and data to a Central Processing Unit (CPU) in a dedicated standalone environment, commonly known as an embedded system. The initial startup of a CPU is known as the boot sequence. One mechanism for the boot sequence is the coupling of a non-volatile memory device such as flash memory to the CPU through a shared address/data bus. Once the CPU is booted and operating, existing networking protocols allows the processor to access and download files which may reside on a device which is attached to the network. One protocol for communicating through a network

1 is the Internet Protocol (IP), as described in the standards
2 of the Internet Engineering Task Force (IETF at
3 www.ietf.org). The Internet Protocol is known as a layer 3
4 protocol on the ISO layer model, and provides for a
5 connection oriented packet transport interface which
6 includes mechanisms for detecting lost packets and
7 requesting retransmission as well as managing timeouts and
8 transmission retry requests. One such IP protocol for
9 transmitting files over a network is the File Transfer
10 Protocol (FTP), as described in RFC-959 found on
11 www.ietf.org/rfc/rfc959.txt. A simpler file transfer
12 protocol is the Trivial File Transfer Protocol (TFTP) also
13 known as RFC1350, described in www.ietf.org/rfc/rfc1350.txt,
14 whereby a network device is attached to a network and
15 requests data using the TFTP protocol which includes
16 verification of transmission of data in a form much simpler
17 than FTP or IP, and TFTP shares the characteristics of
18 acknowledging received data as well as many other error and
19 timeout conditions. A mechanism for booting a device over a
20 network is described in RFC951 (1985), commonly known as the
21 bootstrap protocol, or BOOT-P, and is used for assigning a
22 network address and booting a device from a network. Both
23 of these protocols require the network device have a layer 3
24 (IP) address and be capable of layer 3 (IP) communications,
25 including retransmission. It is desired for a wireless

1 device to utilize wireless layer 2 (MAC) frames, and to
2 operate without an IP address, and without the IP stack
3 being present during the downloading of data from a host.
4 It is further desired to transfer data using fixed length
5 frames without retransmissions requests as used in layer 3
6 protocols.

7 Figure 1 shows a typical prior art embedded wireless
8 system 10, which includes a CPU 12, static random access
9 memory (SRAM) 14, dynamic random access memory (DRAM) 16,
10 all sharing a high speed bus 28, 30, and coupled to a bridge
11 22. The bridge 22 couples access from the master devices
12 such as the CPU 12 on the high speed bus 28, 30 to the long
13 access time devices on the low speed bus 32, 34. The low
14 speed bus 32,34 devices include the bootrom 26, wireless
15 front end 18, and any miscellaneous devices 24 such as real
16 time clocks (not shown), and other devices which are not
17 time-critical for operation of the CPU 12. The high speed
18 bus 28, 30 is generally lightly loaded to enable the highest
19 operating speed of the CPU 12 to occur with frequently
20 accessed devices sharing a high speed bus 28, 30. Accesses
21 to the slower devices on the low speed bus 32, 34 from the
22 CPU 12 are performed through the bridge 22. A typical
23 embedded system has all of its operating software stored on
24 the bootrom 26, such that when the CPU 12 starts, it reads
25 data directly from the bootrom 26, and thereafter copies

1 data and data structures from the bootrom 26 into the
2 memories 14 and 16.

3 In a wireless portable device, the boot program
4 executed by the CPU is typically stored in non-volatile
5 memory such as flash memory bootrom 26. The flash memory
6 typically also contains the entire operating system
7 contents, which is copied from slow flash memory 26 to the
8 high speed SRAM 14 or DRAM 16. Even a minimal operating
9 system for a wireless device typically includes a bootloader
10 which contains the CPU reset vectors, a kernel which
11 provides basic operating system functionality and a
12 scheduler which schedules the various threads and tasks.
13 One such thread is typically a TCP stack, which provides the
14 functionality of the IP (Internet Protocol) layer required
15 by the WFE 18 which is only sending and receiving layer 2
16 MAC frames, and higher layer frames such as IP are handled
17 by operating system software. In a portable wireless device
18 10, the battery life is often governed in part by the size
19 of such non-volatile memory devices. In addition, once the
20 contents of the flash memory device 26 is read by the CPU
21 12, the flash memory device 26 is no longer required.

22 It is desired to reduce the size of the non-volatile
23 flash memory device by providing minimal functionality to
24 the CPU from local non-volatile memory 26, and using the
25 wireless front end 18 to download the operating system from

1 a remote host. In this manner, a smaller non-volatile
2 memory device 26 may be used, which reduces power
3 dissipation and cost. The use of a smaller bootrom is
4 possible because the bootloader is generally a smaller image
5 than the operating system, which can be stored on a remote
6 server.

7 8 9 10 11 Objects of the Invention

12 A first object of the invention is an apparatus for
13 using a ROM controller and a ROM in conjunction with a
14 bridge and a DMA controller to transfer data from a ROM to a
15 static memory.

16 A second object of the invention is an apparatus for
17 using a ROM in conjunction with a DMA controller to transfer
18 data from a ROM to a memory.

19 A third object of the invention is an apparatus for
20 using a memory in conjunction with a CPU and a wireless
21 front end to transfer data from a wireless host to local
22 memory.

23 A fourth object of the invention is an apparatus for
24 using a wireless host and a transfer protocol for sending
25 data from a wireless host to a local memory.

1 A fifth object of the invention is a process for
2 loading an operating system with a first step of loading a
3 SRC, DST, and LENGTH from a ROM to a static RAM, a second
4 step of copying a ROM to a static RAM, and a third step of
5 downloading an operating system from a host server.

6 A sixth object of the invention is a process for
7 transmitting an operating system to an image, said process
8 including the steps of a client sending a download request,
9 a server responding to said download request with an
10 original and duplicate packet, each original and duplicate
11 packet having a sequence number, and transmitting said
12 original and duplicate packet with a sequence number until
13 the download is complete, thereafter sending a "done" packet
14 and a duplicate "done" packet to indicate completion of the
15 download.

16 17 Summary of the Invention

18 A wireless receiver 100 comprises a ROM 116 (Read Only
19 Memory), a ROM Controller 114 coupled to the ROM 116 and
20 also to the low-speed data bus 123 having an address 122 and
21 data 124, a bridge 110 coupling the low speed data bus 123
22 to a high-speed data bus 119 having an address 118 and data
23 120, a DMA (Direct Memory Access) controller 126 coupled to
24 the high speed data bus 119, along with static memory 104,
25 dynamic memory 106, and a Wireless Front End (WFE) 108

1 coupled to the low speed data bus 123. The wireless front
2 end 108 is coupled to a remote wireless host 128 which
3 contains executable operating system code for use by the
4 wireless receiver 100. Upon power-up, the ROM controller
5 114 reads the three data values SRC (Source), DST
6 (Destination), and LENGTH from the ROM 116 and translates
7 these three values to the address of the SRC, DST, and
8 LENGTH registers of the DMA controller 126. The ROM
9 Controller 114 resides on the low-speed bus 123, and the DMA
10 controller 126 resides on the high speed bus 119. The
11 Bridge 110 automatically couples and translates addresses
12 and data from the low speed bus 123 to the high speed bus
13 119 to enable this transfer of data. Once the DMA
14 controller 126 has these three values, it automatically
15 begins a Direct Memory Access sequence, whereby it transfers
16 a LENGTH number of bytes of data specified by the contents
17 of the SRC address to the DST address. If the SRC address
18 points to the ROM contents as accessed by the ROM
19 controller, and the DST points to the memory such as SRAM
20 104, the data is automatically copied from ROM 116 to SRAM
21 104. The CPU 102 remains in an inactive reset state during
22 this transfer. Once the CPU 102 is taken out of reset, it
23 begins executing the code that was earlier copied from ROM
24 116 into memory 104, which also contains the bootup sequence
25 sufficient to begin the downloading of code from the

1 wireless host 126. The CPU 102 requests the balance of code
2 to be downloaded from the wireless front end 108 which is
3 coupled to a wireless host 128, and it is copied into DRAM
4 106. When the download is completed, the CPU has the
5 operating system image required for full operation.

6 7 8 Brief Description of the Drawings

9 Figure 1 shows the block diagram for a prior art
10 wireless receiver.

11 Figure 2 shows the block diagram for a wireless
12 receiver according to the present invention.

13 Figures 3a and 3b show the transaction activity on the
14 high speed bus and low speed bus for the wireless receiver
15 boot sequence of figure 2.

16 Figure 4 is a flowchart for the client download
17 process.

18 Figure 5 is a flowchart for the server download
19 process.

20 21 22 Detailed Description of the Invention

23 Figure 2 shows a wireless system 100, which has a high
24 speed bus 119 comprising an address bus 118 and a data bus
25 120, as is known to one skilled in the art. There may also

1 be a low speed bus 123 comprising an address bus 122 and a
2 data bus 124. A bridge 110 couples the high speed bus 119
3 to the low speed bus 123, as is known to one skilled in the
4 art of bridges. The busses are shown with separate address
5 and data, as may be realized in one embodiment, although the
6 busses could also be a single multiplexed address and data
7 bus, such as PCI (www.pci.org), or SysAd bus of R7000
8 (www.pmc-sierra.com). When a device is controlling a bus
9 through the issuance of read or write requests, it is
10 referred to as a "bus master". Bridge 110 is bi-
11 directional, such that bus masters may be located on the low
12 speed bus 123 or the high speed bus 119. On the low speed
13 bus 123, devices which may act as bus masters are ROM
14 controller 114, bridge 110, and wireless front end 108. On
15 the high speed bus 119, devices which may act as bus masters
16 are the CPU 102 and DMA controller 126. The bridge 110 may
17 translate addresses and data from the high speed bus 119 to
18 the low speed bus 123, or it may preserve them, as one
19 skilled in the prior art of big-endian and little-endian
20 data translations, or data bus width adaptations is aware.
21 In the present invention, bridge 110 requires no
22 initialization, and performs the bridging bi-directionally
23 upon power-up. A sequence controller 130 responds to a
24 power-on signal 131, or any signal which indicates the boot
25 sequence is to begin. The sequence controller 130

1 thereafter generates a ROM controller enable 132, which
2 starts a first sequence of events, and generates a CPU_EN
3 signal 134 at a later time.

4 In the first part of the download sequence, upon
5 assertion of ROM controller enable 132, the ROM controller
6 114 becomes bus master, and reads three values SRC, DST,
7 LENGTH from the ROM 116, and places these values on the low
8 speed bus 123 with the address corresponding to the address
9 of the SRC, DST, and LENGTH registers of the DMA controller
10 126. With these three cycles bus-mastered by the Rom
11 Controller 114, the DMA controller is initialized with the
12 SRC address corresponding to the start location of program
13 memory in ROM 116, the DST address corresponding to a high
14 speed memory such as SRAM 104, and the LENGTH of the
15 transfer, indicating how many bytes of data to transfer.
16 Figure 3a shows this transaction with the ROM controller 114
17 as bus master on the low speed bus 123 sending the SRC, DST
18 and LENGTH data to the DMA controller 126 registers DMA-0,
19 DMA-1, and DMA-2 during first sequence 204.

20 In the second part of the download sequence, the DMA
21 controller 126 uses the SRC (DMA-0), DST (DMA-1), and LENGTH
22 (DMA-2) register values transferred from the earlier
23 sequence to automatically transfer the balance of the data
24 from ROM 116 to memory, shown as SRAM 104, although it would
25 also be possible to copy data to the DRAM 106 by changing

1 the DST address from that of the SRAM 104 to a range
2 occupied by DRAM 106. This is shown in figure 3a as sequence
3 206, which transfers LENGTH bytes of data from the ROM to
4 the SRAM 104. Following the last transfer of data from ROM,
5 the DMA controller 126 removes itself as bus master of the
6 low speed bus 123 and high speed bus 119. At some point
7 thereafter, the sequence controller 130 asserts CPU enable
8 134, which may be achieved by de-asserting a CPU reset line
9 (not shown), as is typically done in the prior art of
10 resetting the processor of figure 1.

11 The third part of the download sequence begins upon
12 assertion of CPU enable 134, and the CPU 102 begins
13 executing instruction cycles from the program data placed in
14 SRAM 104 by the DMA controller 126 from the second sequence
15 previously described. The third part of the sequence 208 is
16 also shown in figure 3b, where the CPU boots, initializes,
17 and starts downloading additional code for the entire
18 operating system from a wireless host such as host 128 of
19 figure 2. The download sequence uses the redundant
20 transmission of data packets, which are reassembled into the
21 complete code block transfer.

22 The flowchart for the client download process 300 is
23 shown in figure 4. The client download process starts 302
24 at step 304, whereby the SRC, DST, and LENGTH are written
25 from the ROM to the DMA controller by the ROM controller,

1 and step 304 corresponds to first sequence 204 of figure 3a.
2 The second step 306 of the download process is the copying
3 of data from the ROM of length LENGTH to the SRAM which is
4 addressed by the DST value written in the first step. The
5 second step 306 corresponds to second sequence 206 of figure
6 3a. The third step of figure 4 corresponding to the CPU
7 download 208 of figure 3b comprises the steps from 308
8 through 326 of figure 4. In step 308, the CPU enable signal
9 has been unasserted, and the CPU boots from the SRAM
10 contents, which contains a minimum image required for
11 booting and the download which occurs in the following
12 steps. Once the CPU is booted and the wireless front end is
13 initialized to send and receive wireless packets in step
14 308, step 310 is performed where a download server is
15 located and the client authenticates itself to the download
16 server by presenting a MAC address, or any method of
17 authentication which allows the download server to determine
18 that a particular client should receive download code. Once
19 the server location and client authentication step 310 is
20 completed, the client sends a "download request" packet in
21 step 312. Each packet received from the wireless front end
22 includes a sequentially increasing "TX_Seq_Num", which is
23 the sequence number of the packet sent by the download
24 server and received by the client. Each download data
25 packet comprises an original packet and a duplicate packet,

1 where both packets contain the same sequence number
2 Tx_Seq_Num. The redundant sending of identical packets
3 reflects the unique wireless operating condition that each
4 packet is a separate receive event, subject to unique
5 channel bit error rate degradation of the communication
6 channel. While two packets are shown, it is possible to
7 transmit any number of redundant packets. For the case of
8 two packets, if the rate of packet corruption or loss is
9 $1/n$, the sending of a redundant packet reduces the error
10 rate to $1/n^2$. The client does not confirm receipt of
11 packets, as is known to one skilled in the art of UDP
12 packets. The client maintains a "RX_Seq_Num" value, which
13 is incremented and compared to the TX_Seq_Num contained in
14 the received original or duplicate packet. If the original
15 packet is received, the duplicate packet is discarded. Step
16 314 shows the initialization of the receive packet sequence
17 number Rx_Seq_Num. The Tx_Seq_Num contained in each
18 received packet 316 is compared to the Rx_Seq_Num value to
19 determine if it is a duplicate 318, and discarded if so. If
20 it is the first-received packet with the proper Rx_Seq_Num,
21 the Rx_Seq_Num is incremented by one in step 320, and if it
22 is the second-received packet with the same Rx_Seq_Num, the
23 duplicate packet is discarded in step 318. If a gap in the
24 Rx_Seq_Num of received packets is detected in step 322,
25 indicating that both an original and duplicate packet were

1 both lost, the "image download request" packet is
2 transmitted, starting the process over at step 312. The
3 final packet received from the host is the "done" packet,
4 indicating completion of transmission 324 and end of the
5 process 326. If the packet is not a "done" packet, the
6 process continues receiving code image packets in step 316.

7 The server download process is shown in figure 5. The
8 process enters at step 500 and waits for a download request
9 502, which is accompanied by a layer 2 MAC address, or any
10 other authentication credentials which may be presented.
11 These credentials are examined in step 504, and upon
12 authentication, the host determines the number of packets to
13 be sent and initializes Num_Packets, which indicates the
14 number of packets to be sent. The transmit sequence number
15 Tx_Seq_Num is initialized in step 508, and the transmission
16 of packets starts in step 510. The transmitter sequence
17 number Tx_Seq_Num is included in an original packet 510, as
18 well as a duplicate packet 512 which is sent an interval of
19 time later, after which the sequence number Tx_Seq_Num is
20 incremented in step 514. The first interval of time between
21 the transmission of an original and duplicate packets may
22 be varied from 1us to 1s, and the second interval of time
23 from a duplicate packet to the following original packet may
24 be less than the first interval. In this manner, other
25 transmission activity on the shared wireless channel may

1 occur without collision between senders, and the likelihood
2 of packet loss is reduced. If a download request is
3 received from the client in step 516, this usually indicates
4 a packet was lost during reception, and the entire download
5 process is started again at step 506. If the Tx_Seq_Num is
6 equal to the Num_Packets in step 518, this indicates that
7 all of the packets have been transmitted, and the process is
8 completed. The completion is made known to the client by
9 sending a DONE packet in step 520, and the process exits in
10 step 522.

11 Figure 5 shows a download process from a server whereby
12 each data includes an original packet and a duplicate
13 packet, and the transmitter continues until the last packet,
14 which is a DONE packet, while the receiver examines each
15 packet in sequence, until it reaches the DONE packet, and
16 issues a new download request if it detects any missing
17 packets. There are other ways of accomplishing the download
18 involving original and duplicate packets. In another
19 embodiment, the download comprises all original packets,
20 each with an increment Tx_Seq_Num, followed by the DONE
21 packet, followed by the duplicate packets, each with an
22 increment Tx_Seq_Num, followed by the DONE packet. In this
23 manner, the original and duplicate packets are transmitted,
24 although in a non-interleaved manner, whereby the figure 5
25 download interleaves the original packet and duplicate

1 packet, each with the same Tx_Seq_Num and data, and the
2 alternate embodiment sends all original packets and the done
3 packet, followed by the duplicate packets and the done
4 packet. In both schemes, the original and duplicate packets
5 carry the same Tx_Seq_Num and data, but are sent in
6 different sequences.

7 The described processor operating system download
8 process and apparatus may be realized in many different ways
9 in accordance with the invention, and is not to be
10 restricted to the specific embodiments shown as examples.
11 As is known to one skilled in the art, address busses such
12 as address bus 118 and address bus 122 of figure 2 are used
13 to uniquely select devices attached to the address bus, and
14 within those devices responding to applied addresses, the
15 address is further used to uniquely select register or
16 memory locations within those devices. While the LENGTH
17 field may be used to transfer a series of adjacent, or
18 contiguous, data values, it is also possible to transfer any
19 arrangement of contiguous, or non-contiguous values, such as
20 described in the second part of the download process in 206
21 of figure 3a, or 306 of figure 4.

22

23

24

1

2

3

4

5